Министерство образования Республики Беларусь
БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

ТЕСТИРОВАНИЕ И ОТЛАДКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
ДЛЯ ВЫПОЛНЕНИЯ КОНТРОЛЬНОЙ РАБОТЫ
для студентов специальности
1-40 01 01 «Программное обеспечение информационных технологий»
заочной формы обучения

Учебное электронное издание

М и н с к 2 0 1 0

УДК 004.434

А в т о р :
Ю.Б. Попова

Р е ц е н з е н т ы :
Г.И. Гульков, заведующий кафедрой «Электропривод и автоматизация промышленных установок и технологических комплексов» БНТУ, к.т.н., доцент;
И.Л. Ковалева, заместитель декана ФИТР БНТУ, к.т.н., доцент.

Методические указания для выполнения контрольной работы по дисциплине «Тестирование и отладка программного обеспечения» для студентов специальности 1-40 01 01 «Программное обеспечение информационных технологий» заочной формы обучения содержат необходимый теоретический материал, задания для выполнения контрольной работы и некоторые примеры.

Белорусский национальный технический университет
пр-т Независимости, 65, г. Минск, Республики Беларусь
Тел. (017) 293-95-64
Регистрационный № БНТУ/ФИТР49-11.2010

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 4

1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ 5
1.1. Различные подходы к тестированию (черный ящик, белый ящик) 6
1.2. Смежные вопросы тестирования 7
1.3. Требования к программному продукту и тестирование 8
1.4. Модульное тестирование 9
1.4.1. Модульное тестирование и его задачи 9
1.4.2. Обзоры 10
1.4.3. Принципы тестирования структуры программных модулей 12
1.4.4. Способы тестирования взаимодействия модулей 13
1.4.5. Стратегии выполнения пошагового тестирования 14
1.4.6. Объектно-ориентированное тестирование 15

2. ЗАДАНИЯ К КОНТРОЛЬНОЙ РАБОТЕ 19
2.1. Задание № 1. Разработка требований к программному продукту 19
2.2. Задание № 2. Модульное тестирование 19

ЛИТЕРАТУРА 21

ПРИЛОЖЕНИЯ 22
ПРИЛОЖЕНИЕ А. Образец требований 22
ПРИЛОЖЕНИЕ Б. Образец оформления титульного листа 34
ПРИЛОЖЕНИЕ В. Содержание отчета 35

ВВЕДЕНИЕ

Контрольная работа по дисциплине «Тестирование и отладка программного обеспечения» направлена на изучение основ процесса жизненного цикла разработки программных продуктов. Наибольшее внимание уделяется этапу разработки требований к программному продукту и модульному тестированию.

Контрольная работа состоит из двух заданий. Для их выполнения необходимо воспользоваться написанной ранее программой для других дисциплин, например, курсовой работой по дисциплине «Системы управления базами данных». Готовая программа позволит не тратить время на разработку нового программного продукта, а направить все усилия на его тестирование.

Первое задание включает в себя разработку требований к программному продукту. Требования к программному продукту – это документ, в котором подробно описывается весь функционал программы. Этим документом пользуются тестировщики при разработке тестовых случаев для функционального тестирования. Процесс функционального тестирования вынесен в лабораторный практикум, поэтому требования к программе пригодятся для его выполнения.

Второе задание заключается в проведении модульного тестирования разработанной программы. Модульное тестирование – это процесс поиска ошибок в программных модулях. Этим видом тестирования занимается автор кода (в отличие от функционального тестирования, которым занимается тестировщик).

Данные методические указания для выполнения контрольной работы по дисциплине «Тестирование и отладка программного обеспечения» содержат теоретические основы для выполнения задания, перечень заданий, список литературы, образец требований к программному продукту (прил. А), образец титульного листа для отчета по контрольной работе (прил. Б) и содержание отчета по контрольной работе (прил. В).

1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ

Тестирование программ зародилось практически одновременно с программированием. Машинное время стоило дорого, поэтому предприятия с повышенными требованиями к надежности программ (например, авиакосмическая промышленность) стали активно разрабатывать методики тестирования.

Долгое время было принято считать, что целью тестирования является доказательство отсутствия ошибок в программе. Однако этот тезис не выдерживает критики, т.к. полный перебор всех возможных вариантов выполнения программы находится за пределами вычислительных возможностей даже для очень небольших программ. Поэтому никакое тестирование не может гарантировать отсутствия ошибок.

Поэтому со временем понимание целей тестирования изменилось. Сегодня общепринятым определением тестирования считается следующее, данное Гленфордом Майерсом [1]: "*Тестирование* – это процесс выполнения программ с целью обнаружения ошибок".

До начала 80-х годов процесс тестирования программного обеспечения (ПО) был разделен с процессом разработки: вначале программисты реализовывали заданную функциональность, а затем тестировщики приступали к проверке качества созданных программ. Однако такой подход создает множество проблем. Например, разработка программ может оказаться достаточно длительной (скажем, несколько месяцев) – чем в это время должны заниматься тестировщики?

Другая, более серьезная проблема заключается в плохой предсказуемости результатов такого процесса разработки. Ключевой вопрос таков: сколько времени потребуется на завершение продукта, в котором существует 500 известных ошибок? На самом деле, предугадать это совершенно невозможно, так как разные ошибки будут требовать разного времени на исправление, а исправление известных ошибок будет неизбежно связано с внесением новых. Существует следующая мрачная статистика: даже однострочное изменение в программе с вероятностью 55 % либо не исправляет старую ошибку, либо вносит новую. Если же учитывать изменения любого объема, то в среднем менее 20 % изменений корректны с первого раза.

В связи с этим в 90-х годах появилась другая методика разработки, которую вслед за Microsoft'ом называют *zero-defect mindset*. Основная идея этой методики заключается в том, что качество программ проверяется не *post factum*, а постоянно в процессе разработки. Например, программист не может перейти к разработке новой функциональности, если существуют известные ошибки высокого приоритета в частях, разработанных им ранее.

При такой постановке вопроса тестирование становится центральной частью любого процесса разработки программ. С другой стороны, *zero-defect mindset* предъявляет существенно более высокие требования к квалификации инженера тестирования: теперь в сферу его ответственности попадает не только функциональное тестирование, но и организация процесса разработки (процесс ежедневной сборки, участие в инспекциях, сквозных просмотрах и обычное чтение исходных текстов тестируемых программ). Поэтому идеальной кандидатурой на позицию тестировщика становится наиболее опытный программист в команде. Такой специалист мог бы не просто существенно улучшить качество создаваемого продукта, но и передать свой опыт младшим товарищам.

1.1. Различные подходы к тестированию (черный ящик, белый ящик)

Долгое время основным способом тестирования было тестирование *методом "черного ящика"* – программе подавались некоторые данные на вход и проверялись результаты в надежде найти несоответствия. При этом, как именно работает программа, считается несущественным. Следует отметить, что даже при таком подходе необходимо иметь спецификацию программы для того, чтобы было с чем сравнивать результаты.

Этот подход до сих пор является самым распространенным в повседневной практике, но у него есть целый ряд недостатков. Во-первых, таким способом невозможно найти взаимоуничтожающихся ошибок, во-вторых, некоторые ошибки возникают достаточно редко (ошибки работы с памятью) и потому их трудно найти и воспроизвести и т.д.

В связи с этим появились методы тестирования, которые изучают не только внешнее поведение программы, но и ее внутреннее устройство (исходные тексты). Такие методики обобщенно называют *тестированием "белого ящика"*. К ним относятся: обзоры кода, инспекции, аудит, критический анализ и т.д. Основной трудностью подобных методов является сложность отслеживания вычислений времени выполнения.

Внимательное изучение этих методов тестирования показывает, что они дополняют друг друга, то есть различные методы находят разные ошибки. Поэтому наиболее эффективные процессы разработки ПО используют некоторую комбинацию методик "черного ящика" и "белого ящика".

В табл. 1.1 приводятся показатели минимальной, средней и максимальной эффективности различных методов тестирования:

Таблица 1.1

Показатели эффективности различных методов тестирования

Название методики	Минимальная эффективность	Средняя эффективность	Максимальная эффективность
Персональные просмотры проектных документов	15 %	35 %	70 %
Неформальные групповые просмотры	30 %	40 %	60 %
Формальные просмотры проектных документов	35 %	55 %	75 %
Формальные инспекции кода	30 %	60 %	70 %
Моделирование и прототипирование	35 %	65 %	80 %
Проверка за партой	20 %	40 %	60 %
Тестирование модулей	10 %	25 %	50 %
Функциональное тестирование	20 %	35 %	55 %
Комплексное тестирование	25 %	45 %	60 %
Тестирование в реальных условиях	35 %	50 %	65 %
Применение всех перечисленных методик тестирования	**93 %**	**99 %**	**99 %**

Из таблицы видно, что тестирование максимально эффективно в тех случаях, когда программа проверяется не только путем запуска, но и путем чтения, статических проверок и т.п. Поэтому классическое определение Майерса, приведенное выше, оказывается слишком узким и не охватывающим всех аспектов современного тестирования. В связи с этим будем пользоваться другим определением тестирования:

Определение: *Тестирование* – это любая деятельность, направленная на обнаружение ошибок в программном продукте.

1.2. Смежные вопросы тестирования

Об экономической стороне тестирования. Тестирование само по себе является затратной деятельностью, отнимающей время и деньги. Поэтому в большинстве случаев разработчики программного обеспечения заранее формулируют какой-либо критерий качества создаваемых программ (определяют, так называемую, *планку качества*), добиваются выполнения этого критерия и после этого прекращают тестирование и выпускают продукт на рынок. Такая концепция получила название *Good Enough Quality* (достаточно хорошее ПО), в противовес более очевидной концепции *Best Possible Quality* (максимально качественное ПО).

К сожалению, принцип *Good Enough Quality* зачастую понимают неправильно, ближе к формулировке *Quality - If Time Permits* (качество - если будет время). Конечно, выпуск плохо протестированного продукта из-за недостатка времени – это плохая практика. Опыт показывает, что пользователи склонны со временем забывать даже значительные задержки с выпуском продукта, но плохое качество выпущенного продукта запоминается на всю жизнь.

На самом деле, *Good Enough Quality* – это просто поиск разумного компромисса между затратами на тестирование, длительностью разработки продукта и его качеством. Ничего умнее пока не придумали.

Психологические аспекты тестирования. Тестирование принципиально отличается от программирования по своим психологическим характеристикам. Дело в том, что программирование носит конструктивный характер, а тестирование ПО деструктивно по своей природе. Можно сказать, что программист создает, строит, а тестировщик отыскивает недостатки этих построений. Поэтому программисты так часто не замечают очевидных ошибок в своих программах: к своему творчеству трудно относиться критично.

Все это не значит, что тестирование "хуже", чем программирование, или что в процессе тестирования нет места творчеству – просто эти виды деятельности отличаются коренным образом, и для тестирования требуется иной склад характера, чем для программирования. Следовательно, программист не должен тестировать свои собственные программы – для этого необходим другой человек. Более того, программист не должен тестировать даже чужие программы! Дело в том, что программист потратит какое-то время, перенастраиваясь на новую задачу. Даже переключение с одной программистской задачи на другую требует обычно от 10 минут до получаса. Понятно, что переключение с одного образа мышления на другой отнимет существенно больше времени – возможно, даже день или два.

Таким образом, принято считать, что команда разработчиков не должна совпадать с командой инженеров тестирования, но при этом разработчики и тестировщики должны постоянно взаимодействовать друг с другом для достижения приемлемого качества окончательного продукта.

1.3. Требования к программному продукту и тестирование

Разработка любого программного продукта начинается с выявления требований к этому продукту. Проводятся встречи, интервью с заказчиками, в результате чего составляется документ, в котором отражены все требования к продукту. Там описываются как функциональные (что должна делать программа), так и нефункциональные (например, на каком оборудовании

должна работать программа) требования. Пример требований приведен в приложении А.

1.4. Модульное тестирование

1.4.1. Модульное тестирование и его задачи

Модульное тестирование – вид тестовой деятельности, при которой проверке подвергаются внутренние рабочие части программы, элементы или модули независимо от способа их вызова. Под модулем принято понимать программу или ограниченную часть кода с одной точкой входа и одной точкой выхода, которая выполняет одну и только одну первичную функцию. Этот тип тестирования, как правило, осуществляется программистом, а не тестировщиков, поскольку для его проведения необходимы доскональные знания внутренней структуры и кода программы. Такое тестирование не всегда выполняется просто, например, в случае, если прикладная программа не имеет четко определенной архитектуры с компактным кодом, и для его проведения может понадобиться разработка модулей тестовых драйверов или средств тестирования.

Тестировать свои собственные продукты – это весьма трудное занятие для разработчиков, поскольку они вынуждены изменить свою точку зрения или отношение, перейдя от роли создателя в роль критика. Многие разработчики не любят тщательно тестировать свои программные продукты, так как считают скучным и даже угнетающим наблюдать за тем, как прикладная программа справляется с возложенными на нее нагрузками. Другие не имеют ничего против подобного подхода и прекрасно выполняют такого рода работу. Такой подход, несомненно, приводит к обнаружению ошибок на более ранних этапах разработки программного продукта, а это в свою очередь способствует снижению стоимости ошибки. Даже самый опытный программист допускает ошибки. Некоторые исследования показывают, что плотность распределения дефектов находится в диапазоне от 49,5 до 94,6 на тысячу строк кода. Поэтому каждый разработчик должен самостоятельно тестировать свои "произведения" с максимальной тщательностью, прежде чем передать рабочий продукт независимому испытателю, т.е. тестировщику.

Методы, используемые при модульном тестировании, различаются по следующим признакам [5]:

- по степени автоматизации – ручные и автоматизированные методы;
- по форме представления модуля – символьное представление (на языке программирования) или в машинном коде;

- по компонентам программы, на которые направлено тестирование – структура программы или преобразование переменных;
- статические или динамические методы.

В [5] отмечается, что в первую очередь следует тестировать структуру программы, так как операторы анализа условий составляют в среднем 10–15 % от общего числа операторов программы. Искажение логики работы программы приводит к серьезным ошибкам. К тому же данный вид тестирования имеет наилучшие показатели «эффективность/стоимость». Там же предлагается следующая методика тестирования модулей: последовательно проводить различные виды тестирования, начиная с самых простых:

- ручное тестирование (работа за столом);
- символическое тестирование (инспекции, сквозные просмотры);
- тестирование структуры;
- тестирование обработки данных;
- функциональное тестирование (сравнение со спецификацией, взаимодействие с другими модулями).

Из приведенных выше видов тестирования, ручное и символическое относятся к статическому тестированию, которое проводится без запуска программы. Эти два вида, как правило, основаны на обзоре программного кода, только первый проводится автором-разработчиком, а второй - сторонними специалистами (другими разработчиками, инженерами по качеству или приглашенными инспекторами).

Последние три из перечисленных видов относятся к динамическому тестированию, и проведение каждого из них состоит из следующих этапов:

- планирование тестирования (разработка тестов, формирование контрольных примеров);
- собственно тестирование;
- обработка результатов тестирования.

1.4.2. Обзоры

Как уже было сказано выше, обзоры являются основой статического тестирования и способны нейтрализовать действие человеческого фактора при выполнении поставленных задач. При модульном тестировании можно выделить следующие положительные моменты обзоров (учитывая, что оно может проводиться как самим разработчиком, так и сторонними специалистами) [2, 3]:

1. Обзоры позволяют обнаружить посторонние элементы, что нельзя сделать в ходе традиционного тестирования. Обзоры могут следовать по всем

выполняемым ветвям разрабатываемого ПО, а с помощью тестирова-ния невозможно проверить все ветви. В результате этого редко проходимые ветви часто приводят к появлению ошибок.

2. Разные точки зрения, личностные качества и жизненный опыт помогают при обнаружении всех видов проблем, которые незаметны при поверхностном взгляде.

3. Разработчики могут больше внимания уделять творческому аспекту процесса разработки, зная, что их эксперты участвуют в проекте и действуют в качестве "подстраховки", и всегда готовы увидеть реальную картину.

4. Происходит разделение труда, благодаря чему рецензенты могут сконцентрироваться на этапе обнаружения проблем.

5. Распространение информации и обучение происходят тогда, когда разработчики встречаются при проведении обзора. Люди быстро воспринимают и распространяют хорошие идеи, которые они замечают в работе других пользователей. По словам некоторых экспертов – это самый важный результат выполнения обзора.

6. Возрастает степень согласованности работы между членами команды. Происходит установление фактических групповых норм, что облегчает понимание сути программных продуктов и их сопровождение.

7. Менеджеры проекта могут получить представление о действительном состоянии разрабатываемых продуктов.

В конечном счете, в результате обзора программные продукты улучшаются в силу следующих причин:

- Проблемы обнаруживаются тогда, когда их можно относительно легко исправить без значительных понесенных затрат.
- Локализация проблем происходит практически в месте их возникновения.
- Инспектируются исходные данные какой-либо фазы с целью проверки их соответствия исходным требованиям или критериям выхода.
- Предотвращается возникновение дальнейших проблем с помощью оглашения решений часто происходящих затруднений.
- Распространяются сведения о проекте, что способствует повышению его управляемости.
- Разработчики обучаются тому, каким образом избежать возникновения дефектов при дальнейшей работе.
- Предотвращается возникновение дефектов в текущем продукте, поскольку процесс подготовки материалов для инспекционной проверки способствует уточнению требований и проекта.
- Обучаются новые участники проекта.

- Менеджерам проекта предоставляются надежные опорные точки и предварительные оценки.
- Облегчается поддержка установленного порядка выполнения проекта и обеспечивается объективная, измеримая обратная связь.

1.4.3. Принципы тестирования структуры программных модулей

Как правило, тестирование структуры программного модуля происходит с помощью методов графического отображения модуля, например, диаграммы Чейпина или Несси Шнайдермана, ориентированные графы Мак-Кейба. Подобно Чейпину, Мак-Кейб применяет следующие конструкции: if-then-else, if then, do-while, case и repeat-until. Эти конструкции изображены на рис. 1 [3].

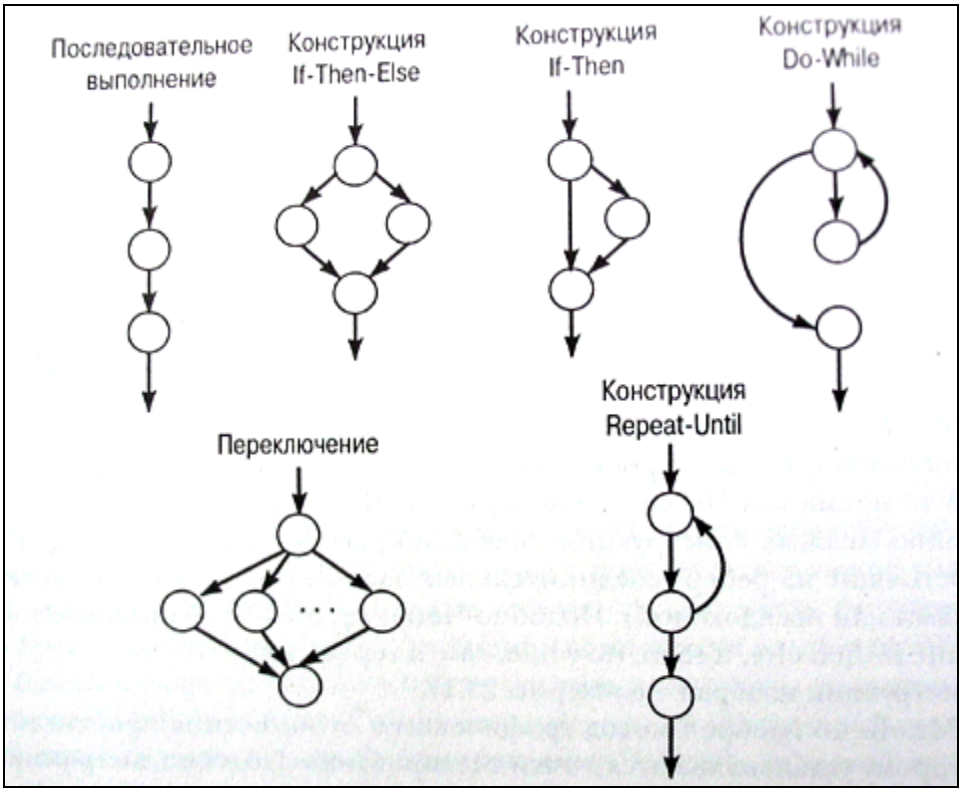

Рис. 1. Графические конструкции для модульного тестирования

Метрический показатель сложности или цикломатическое число G потокового графа определяется по формуле:

$$G = R - V + 2,$$

где R – количество ребер графа;
V – количество вершин графа.

Следует отметить, что данная формула справедлива только для замкнутого графа. Чтобы вычислить этот коэффициент вручную для большой программы, потребуется немало усилий. К счастью, существуют автоматические средства, которые вычисляют метрический показатель сложности Мак-Кейба путем анализа существующего исходного кода.

Метрический показатель сложности не только может пригодиться при установлении проблемных областей, но также может использоваться при создании контрольных примеров. Как только потоковый граф создан, очевидными становятся пути, ведущие через программу, которые предоставляют информацию, необходимую для их выполнения в тесте.

При планировании тестирования структуры программных модулей решаются 2 задачи [5]:

- формирование критериев выделения маршрутов в программе;
- выбор стратегий упорядочения выделенных маршрутов.

Критерии выделения маршрутов в программе могут быть следующими:

- минимальное покрытие графа программы;
- маршруты, образующиеся при всех возможных комбинациях входящих дуг.

Стратегия упорядочения маршрутов выбирается по:

- длительности исполнения и числу команд в маршрутах. Такая стратегия выбирается при тестировании программ вычислительного характера.
- количеству условных переходов, определяющих формирование данного маршрута. Данная стратегия выбирается при тестировании логических программ с небольшим объемом вычислений по вероятности исполнения маршрутов. Применяется в тех случаях, когда сложно оценить вероятности ветвления в условных переходах, а также число исполнений циклов.

1.4.4. Способы тестирования взаимодействия модулей

В [1] автор выделяет два способа проверки взаимодействия модулей:

- монолитное тестирование;
- пошаговое тестирование.

Пусть имеется программа, состоящая из нескольких модулей, представленных на рис. 2.

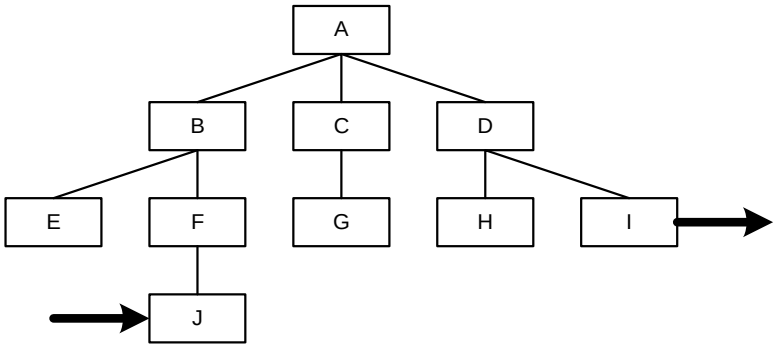

Рис. 2. Схема многомодульной программы

Обозначения на рис. 2 следующие: прямоугольники – это модули, тонкие линии представляют иерархию управления (связи по управлению между модулями), жирные стрелки – ввод и вывод данных в программу.

Монолитное тестирование [1, 5] заключается в том, что каждый модуль тестируется отдельно. Для каждого модуля пишется один модуль-драйвер, который передает тестируемому модулю управление, и один или несколько модулей-заглушек. Например, для модуля B нужны 2 заглушки, имитирующие работу модулей E и F. Когда все модули протестированы, они собираются вместе, и тестируется вся программа в целом.

Пошаговое тестирование предполагает, что модули тестируются не изолированно, а подключаются поочередно к набору уже оттестированных модулей.

Можно выделить следующие *недостатки* монолитного тестирования (перед пошаговым) [5, 6]:

1. Требуется много дополнительных действий (написание драйверов и заглушек).
2. Поздно обнаруживаются ошибки в межмодульных взаимодействиях.
3. Следствие из 2 – труднее отлаживать программу.

К *преимуществам* монолитного тестирования можно отнести:

1. Экономию машинного времени (в настоящее время существенной экономии не наблюдается)
2. Возможность параллельной организации работ на начальной фазе тестирования.

1.4.5. Стратегии выполнения пошагового тестирования

Существуют две принципиально различные стратегии выполнения пошагового тестирования [5]:

- нисходящее тестирование;

- восходящее тестирование.

Нисходящее тестирование начинается с головного модуля, в нашем случае с модуля A. Возникают проблемы: как передать тестовые данные в A, ведь ввод и вывод осуществляется в других модулях? Как передать в A несколько тестов?

Решение можно представить в следующем виде:

а) написать несколько вариантов заглушек B (для каждого теста);

б) написать заглушку B так, чтобы она читала тестовые данные из внешнего файла.

В качестве стратегии подключения модулей можно использовать одну из следующих:

а) подключаются наиболее важные с точки зрения тестирования модули;

б) подключаются модули, осуществляющие операции ввода/вывода (для того чтобы обеспечивать тестовыми данными «внутренние» модули).

Нисходящее тестирование имеет ряд недостатков: предположим, что модули I и J уже подключены, и на следующем шаге тестирования заглушка H меняется на реальный модуль H. Как передать тестовые данные в H? Это нетривиальная задача, потому что между J и H имеются промежуточные модули и может оказаться невозможным передать тестовые данные, соответствующие разработанным тестам. К тому же достаточно сложно интерпретировать результаты тестирования H, так как между H и I также существуют промежуточные модули.

Восходящее тестирование практически полностью противоположно нисходящему тестированию. Начинается с терминальных (не вызывающих другие модули) модулей.

Стратегия подключения новых модулей также должна основываться на степени критичности данного модуля в программе. *Восходящее тестирование* лишено недостатков нисходящего тестирования, однако имеет свой, главный недостаток: рабочая программа не существует до тех пор, пока не добавлен последний (в нашем случае A) модуль.

Выбор одной их двух представленных стратегий определяется тем, на какие модули (верхнеуровневые или нижнеуровневые) следует обратить внимание при тестировании в первую очередь.

1.4.6. Объектно-ориентированное тестирование

С традиционной точки зрения [7] наименьший контролепригодный элемент – это элемент или модуль, который можно протестировать методом "белого ящика". При объектно-ориентированном тестировании этот функциональный элемент не отделяется от своего класса, в котором он

инкапсулирован со своими методами. Это означает, что поэлементное тестирование по существу заменяется классовым тестированием. Однако не следует отклоняться слишком далеко от наших корневых принципов относительно того, что каждый метод класса объектов можно рассматривать как "небольшой элемент", тестирование которого можно выполнять обособленно, применяя для этого методы "белого и черного ящика". Классы объектов необходимо протестировать во всех возможных состояниях. Это означает, что необходимо сымитировать все события, вызывающие изменения состояния объекта.

В традиционном отношении [7] элементы компилируются в подсистемы и подвергаются интеграционным тестам. При объектно-ориентированном подходе не применяется тестирование структурной схемы сверху вниз, поскольку точно не известно, какой класс будет вызван пользователем за предыдущим. Интеграционные тесты заменяются тестированием функциональных возможностей, при котором тестируется набор классов, которые должны реагировать на один входной сигнал или системное событие, или же эксплуатационным тестированием, при котором описывается один способ применения системы, основанный на сценариях случаев эксплуатации.

Тестирование взаимодействия объектов следует по путям сообщения с целью отследить последовательность взаимодействий объектов, которая заканчивается только тогда, когда был вызван последний объект. При этом не посылается сообщение и не вызываются службы любого другого объекта.

Системное, альфа-, бета-тестирование и приемочное испытание пользователем изменяются незначительно, поскольку объектно-ориентированная система проходит эксплуатационные тесты точно также, как и разработанные традиционным способом системы. Это означает, что процесс V&V по существу не изменяется.

Тестирование классов охватывает виды деятельности, направленные на проверку соответствия реализации этого класса спецификации. Если реализация корректна, то каждый экземпляр этого класса ведет себя подобающим образом.

Тестирование классов в первом приближении аналогично тестированию модулей и может проводиться с помощью обзоров (ревизий) и выполнения тестовых случаев.

К недостаткам обзоров (ревизий) относятся:

- возможные ошибки, обусловленные человеческим фактором;
- значительно большие затраты и усилия, нежели регрессионное тестирование.

Тестирование в режиме прогона тестовых случаев лишено указанных выше недостатков, однако необходимо приложить значительные усилия для отбора подходящих тестовых случаев и для разработки тестовых драйверов.

Прежде чем приступать к тестированию класса, необходимо определить, тестировать его в автономном режиме как модуль или каким-то другим способом, как более крупный компонент системы. Для этого необходимо выяснить:

- роль класса в системе, в частности, степень связанного с ним риска;
- сложность класса, измеряемая количеством состояний, операций и связей с другими классами;
- объем трудозатрат, связанных с разработкой тестового драйвера для тестирования этого класса.

Если какой-либо класс должен стать частью некоторой библиотеки классов, целесообразно выполнять всестороннее тестирование классов, даже если затраты на разработку тестового драйвера окажутся высокими.

При тестировании классов можно выделить 5 оцениваемых факторов [9]:

1. **Кто выполняет тестирование.** Как и при модульном тестировании, тестирование классов выполняет разработчик, поскольку он знаком со всеми подробностями программного кода. Основной недостаток того, что тестовые драйверы и программные коды разрабатываются одним и тем же персоналом, заключается в том, что неправильное понимание спецификации разработчиками распространяется на тестовые наборы и тестовые драйверы. Проблем такого рода можно избежать путем привлечения независимых тестировщиков или других разработчиков для ревизий программных кодов.

2. Что тестировать. Тестировать нужно программный код на точное соответствие его требованиям, т.е. в классе должно быть реализовано все запланированное и ничего лишнего. Если для конкретного класса характерны статические элементы, то их также необходимо тестировать. Такие элементы принадлежат самому классу, но не каждому экземпляру.

3. **Когда тестировать.** Тестирование класса должно проводиться до того, как возникнет необходимость его использования. Необходимо также проводить регрессионное тестирование класса каждый раз, когда меняется его реализация. Однако до начала тестирования класса необходимо закодировать класс и разработать тестовые случаи использования класса. Ранняя разработка тестовых случаев позволяет программисту лучше понять спецификацию класса и построить более успешную реализацию класса, которая пройдет все тестовые случаи. Существует даже практика первоначальной разработки тестовых случаев, а затем программного кода. Такой подход направлен на изначально все предусматривающий

программный код. Главное, чтобы этот подход не привел к проблемам во время интеграции этого класса в более крупную часть системы.

4. **Каким образом тестировать.** Тестирование классов обычно выполня-ется путем разработки тестового драйвера. Тестовый драйвер создает экземпляры классов и окружает их соответствующей средой, чтобы стал возможен прогон соответствующего тестового случая. Драйвер посылает одно или большее количество сообщений экземпляру класса (в соответствии с тестовым случаем), затем сверяет результат его работы с ожидаемым и составляет протокол о прохождении или не прохождении теста. В обязанности тестового драйвера обычно входит и удаление созданного им экземпляра.

5. **В каких объемах тестировать.** Адекватность может быть измерена полнотой охвата тестами спецификации и реализации класса. Т.е необходимо тестировать операции и переходы состояний в различных сочетаниях. Поскольку объекты находятся в одном из возможных состояний, эти состояния определяют значимость операций. Поэтому требуется определить, целесообразно ли проводить исчерпывающее тестирование. Если нет, то рекомендуется выполнить наиболее важные тестовые случаи, а менее важные выполнять выборочно.

2. ЗАДАНИЯ К КОНТРОЛЬНОЙ РАБОТЕ

2.1. Задание № 1
Разработка требований к программному продукту

Постановка задачи

Разработать требования к программному продукту, разработанному ранее, например, в рамках курсовой работы по дисциплине СУБД по образцу, приведенному в Приложении А.

2.2. Задание № 2
Модульное тестирование

Постановка задачи

1. Провести обзор разработанного программного кода. В случае обнаружения ошибок составить отчет следующего содержания (табл. 2.1).

Таблица 2.1

Ошибки обзора

Номер ошибки	**Название модуля/функции**	**Описание ошибки**	**Важность ошибки (высокая, средняя, низкая)**	**Ошибка исправлена Да/Нет**
1	Func1()	В первом цикле for не верно указано значение окончания цикла	средняя	Да

2. Для каждого модуля построить графы и вычислить цикломатические числа. Разработать тестовые случаи для каждого графа и представить их в виде табл. 2.2.

Таблица 2.2

Модуль Func1()

G	**Номер сцена-рия**	**Описание прохода**	**Контрольные примеры, позволяющие реализовать описанную ситуацию**	**Тест пройден Да/Нет**
G=2	1	a-b-d-f	Field=4, x=-5	Нет
	2	a-b-e-f	Field=-3, x=-5	Да

3. Провести модульное тестирование согласно составленным тестовым случаям. При необходимости разработать заглушки и привести их в табл. 2.3.

Таблица 2.3

Описание заглушек

Номер заглушки	**Название модуля**	**Назначение заглушки**	**Текст заглушки**	**Тест пройден Да/Нет**

В случае не прохождения тестового случая составить отчет о каждой найденной ошибке в виде табл. 2.4.

Таблица 2.4

Модульное тестирование

Номер ошибки	**Название модуля**	**Описание прохода**	**Контрольные примеры, позволяющие реализовать описанную ситуацию**	**Описание ошибки**	**Важность ошибки**	**Ошибка исправлена Да/Нет**
1	Func1()	a-b-d-f	Field=4, x=-5	Появляется не обработанное сообщение windows	средняя	да

4. Построить схему взаимодействия модулей.

5. Разработать стратегию тестирования взаимодействия модулей и привести ее в виде табл. 2.5.

Таблица 2.5

Взаимодействие модулей

Номер в последова-тельности	**Описание последователь-ности**	**Контрольные примеры, позволяющие реализовать описанную ситуацию**	**Тест пройден Да/Нет**
1	Func1()->func2()	Argument=0	

ЛИТЕРАТУРА

1. Майерс, Г. Искусство тестирования программ / Г. Майерс. – М.: Финансы и статистика, 1982. – 186 с.
2. Канер, С. Тестирование программного обеспечения / С. Канер. – Киев: ДиаСофт, 2001. – 544 с.
3. Калбертсон, Р. Быстрое тестирование / Р. Калбертсон, К. Браун, Г. Кобб. – М.: Вильямс, 2002. – 384 с.
4. Майерс, Г. Надежность программного обеспечения / Г. Майерс. – М.: Финансы и статистика, 1980. – 220 с.
5. Липаев, В.В. Тестирование программных средств / В.В. Липаев. – М.: Финансы и статистика, 1999. – 264 с.
6. Липаев, В.В. Обеспечение качества программных средств. Методы и стандарты / В.В. Липаев. – М.: Финансы и статистика, 2001. – 186 с.
7. Макгрегор, Д. Тестирование объектно-ориентированного программного обеспечения / Д. Макгрегор, Д. Сайкс. – Киев: ДиаСофт, 2002. – 348 с.
8. Шнейдерман, Б. Психология программирования / Б. Шнейдерман. – М.: Радио и связь, 1984. – 304 с.
9. Бейзер, Б. Тестирование «черного ящика» / Б. Бейзер. – Киев: ДиаСофт, 2002. – 326 с.

ПРИЛОЖЕНИЯ

Приложение А

Образец требований

<Система тестирования знаний на основе базы данных в формате XML>

Версия 1.0

ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

Автор: Иванов И.И.

Введение

Целью создания этого документа является определение набора требований к программному продукту на тему “Система тестирования знаний на основе базы данных в формате XML”, именуемому SmartTest. Документ предназначен для разработчиков и тестировщиков ПП. Документ организован таким образом, что позволяет выделять, идентифицировать и выбирать отдельные требования. Требования излагаются на таком уровне детализации, что на их основе разработчики могут создавать программный продукт, а тестировщики – выполнять аттестацию этого продукта. Этот документ предназначен только для внутреннего использования.

Программный продукт SmartTest может использоваться для упрощения процессов тестирования знаний студентов по различным темам, а также на предприятиях для тестирования новых сотрудников при приеме на работу и при повышении квалификации работающих сотрудников.

ПП представляет собой Web-приложение, благодаря чему пользователи могут получить удаленный доступ к данному ресурсу.

А.1. Общее описание

А.1.1. Функции продукта

В этом разделе описываются функциональные высокоуровневые свойства программного продукта SmartTest. Более подробное описание требований находится в разделе 2.0.

А.1.1.1. Работа в роли администратора

ПП SmartTest должен обеспечивать следующие возможности для администрирования:

- средства создания, просмотра, удаления пользователей, а также изменения данных пользователя
- средства добавления, просмотра, удаления, редактирования тестов
- средства просмотра, добавления, вопросов к заданному тесту, редактирования и удаления этих вопросов

- возможность задания максимально допустимого времени ответа на каждый добавляемый вопрос, и одного из трех уровней сложности каждого вопроса.
- возможность выбора использования вариантов ответов для вопроса (при создании вопроса) для последующей программной обработки ответов, или ответа в свободной форме для последующей проверки их вручную.
- возможность создания, редактирования списка вариантов ответа для каждого вопроса, в котором предусмотрено использование вариантов, а также задание варианта ответа, который является правильным

A.1.1.2. Работа в роли клиента

ПП SmartTest должен обеспечивать следующие возможности для клиента:

- возможность регистрации с установлением пароля
- возможность просмотра полного списка тестов, доступ к которым открыт для клиента.
- средства выбора нужной темы и возможность прохождения теста по выбранной теме
- возможность просмотреть свои результаты по окончании теста в виде полного списка, содержащего номер вопроса (без текста вопроса), уровень его сложности, и отметку, правильным ли был ответ.

A.1.2. Пользовательские характеристики

Пользователями описываемого программного продукта могут являться студенты, служащие, работники предприятий и кандидаты на работу, а также руководители предприятий и их подразделений, желающие повысить уровень контроля знаний своих подчиненных и проверить степень усвояемости ими новой информации.

A.1.3. Общие ограничения

Ниже перечислены ограничения, которые могут повлиять на возможности команды разработчиков программного обеспечения (ПО):

- Ограничения, связанные с оборудованием: должна быть возможность эксплуатации программы на архитектуре ПК основанной на процессоре Pentium II и выше;
- Разделение прав пользователей: регистрация пользователей производится при входе в систему;
- Требования, накладываемые языками высокого уровня: в качестве среды программирования выбрать Visual Studio 2003.net, язык C# .

A.1.4. Допущения и зависимости

В этом разделе описаны допущения и зависимости, связанные с программным продуктом SmartTest. С целью упрощения будущих ссылок каждое допущение или зависимость помечается соответствующим заголовком. Перечисленные допущения следует принимать во внимание во время создания конфигураций для тестирования SmartTest.

A.1.4.1. Операционные системы

Предполагается, что пользователь выполняет клиентское приложение на компьютере, работающем под управлением одной из следующих операционных систем: Microsoft Windows XP, Microsoft Windows NT 3/51 или выше , Microsoft Windows 2000.

Предполагается, что серверное приложение выполняется на компьютере, работающем под управлением операционной системы Microsoft Windows NT 3/51 или выше.

А.1.4.2. Браузеры

Предполагается, что пользователь на клиентском компьютере использует один из следующих браузеров: IE 5.5 или выше, Opera 7.02 или выше.

А.1.4.3. Базы данных

Предполагается использование базы данных, содержащей сведения о пользователях, а также информацию о темах теста, вопросах по этим темам, и вариантов ответов на эти вопросы. Также в эту базу данных будет заноситься оценка, выставленная людям, прошедшим тест. База данных хранится в XML-файле.

А.1.4.4. Зависимость от процессора

Приложение не зависит от типа применяемого процессора. Перечисленные ранее допустимые операционные системы могут использоваться на платформах с процессорами x86, RISC, SPARC, Motorola или PPC.

А.2. Специальные требования

В этом разделе представлены детализованные требования, относящиеся к программному продукту SmartTest.

А.2.1. Функциональные требования

А.2.1.1. Пользовательский интерфейс

Пользовательский интерфейс для клиента <название программы> создается с использованием Visual Studio 2003.net.

А.2.1.2. Навигация

Главное меню ПП SmartTest, которое увидит пользователь-администратор, включает следующие пункты:

Тесты
Пользователи

А.2.1.3. Аутентификация пользователя

Данная функциональность позволяет определить роли пользователей: администратора и клиента. Для этого пользователь вводит логин и пароль на странице входа в систему, к.т. отображается сразу после запуска программы. Описание необходимых свойств пользователя и функциональности кнопок – см. таблицы ниже.

Страница входа в систему:

Поле	**Тип**	**Ограничения**	**Описание, параметры, ограничения**
Логин	Text[30]	Уникальный, может содержать буквы, цифры, знак подчеркивания.	Уникальное имя, к.т. используется пользователем для входа в систему.
Пароль	Text[20]	Минимум 5 символов.	Пароль пользователя для входа в систему. Вводимые символы для пароля должны отображаться в виде звездочек.

Кнопки на странице входа в систему:

Название кнопки	Тип	Ограничения	Описание
Войти	Button		Проверяет, есть ли введенные логин и пароль среди данных пользователь, подтвержденных администратором , и если есть, то пользователь оказывается на странице «Тесты». Если нет, то выдается сообщение о том, что пользователь не зарегистрирован или не подтвержден администратором, с предложением зарегистрироваться, или подождать некоторое время.
Регистрация	Гиперссылка		Переводит пользователя на форму регистрации (страница «Информация о пользователе»), и позволяет ему ввести свои данные и зарегистрироваться. Подробное описание – см. соответствующий пункт.

Если в систему входят под ролью администратора, SmartTest позволяет выполнить следующие действия: просмотр, добавление и удаления пользователей, а также изменения данных пользователя; добавление, просмотр, удаление, редактирование тем тестов; просмотр, добавление, вопросов к заданной теме теста, редактирование и удаление этих вопросов; создание, редактирование списка вариантов ответа для каждого вопроса, в котором предусмотрено использование вариантов, а также задание варианта ответа, который является правильным.

Если в систему входят под ролью клиента, SmartTest позволяет выполнить следующие действия: просмотреть доступные темы тестов, выбрать нужную тему теста, пройти тест по заданной теме, и просмотреть свои результаты в виде полного списка, содержащего номер вопроса (без текста вопроса), уровень его сложности, и отметку, правильным ли был ответ.

А.2.1.4. Регистрация пользователя

Любой пользователь имеет возможность зарегистрироваться в программе. Для этого ему нужно на форме регистрации (см. ниже описание страницы «Информация о пользователе») заполнить все поля (все поля являются обязательными) и нажать кнопку «Зарегистрироваться». (Описание необходимых свойств пользователя и функциональности кнопок – см. таблицы ниже.)

После регистрации в системе данные клиента добавляются в базу данных, и этому клиенту присваивается статус «Новый». Пользователь, к.т. находится в статусе «Новый», не может входить в систему, но его данные начинают отображаться в списке пользователей у администратора. Администратор может потом изменить статус пользователя на «Открытый» (Пользователь может проходить тесты) или «Заблокированный» (Пользователь может лишь просматривать темы тестов).

Страница «Информация о пользователе»:

Поле	Тип	Ограничения	Описание, параметры, ограничения
Логин	Text[30]	Уникальный, может содержать буквы, цифры, знак подчеркивания.	Уникальное имя, к.т. используется пользователем для входа в систему.
Имя	Text[50]	Может содержать только буквы	Имя пользователя
Фамилия	Text[50]	Может содержать только буквы	Фамилия пользователя
Роль	DropDownList. Может быть «Клиент», «Администратор»		По умолчанию «Клиент»

E-mail		Формат адреса электронной почты	Адрес электронной почты клиента
Пароль	Text[20]	Минимум 5 символов.	Пароль пользователя для входа в систему. Вводимые символы для пароля должны отображаться в виде звездочек.
Подтверждение пароля	Text[20]	Минимум 5 символов. Значение должно быть равным значению поля «Пароль»	Пароль пользователя для входа в систему. Вводимые символы для пароля должны отображаться в виде звездочек.

Кнопки на странице «Информация о пользователе»:

Название кнопки	**Тип**	**Ограничения**	**Описание**
Зарегистрироваться	Button		Сохраняет данные пользователя в БД, присваивая пользователю статус «Новый».
Отмена	Button		Отменяет регистрацию и возвращает на страницу входа в систему.

А.2.1.5. Просмотр информации о пользователях. Изменение статуса пользователя

Администратор имеет возможность просмотра информации обо всех зарегистрированных пользователях системы. Для просмотра нужно открыть страницу «Пользователи системы», выбрав соответствующий пункт главного меню. Подробное описание страницы «Пользователи системы» - см. таблицы ниже.

Просмотреть более подробные данные о пользователе можно, выделив одного пользователя и нажав кнопку «Редактировать пользователя». Появится страница «Информация о пользователе», которую можно там же и отредактировать. (Подробности редактирования данных пользователя – см. соответствующий пункт.)

Чтобы изменить статус пользователя на «Открытый» (Тогда пользователь сможет проходить тесты) нужно выбрать пользователя (или нескольких пользователей) и нажать кнопку «Разблокировать» на странице «Пользователи системы».

Чтобы изменить статус пользователя на «Заблокированный» (Тогда пользователь сможет лишь просматривать темы тестов) нужно выбрать пользователя (или нескольких пользователей) и нажать кнопку «Заблокировать» на странице «Пользователи системы».

Страница «Пользователи системы»:

Поле	**Тип**	**Обязательное**	**Описание, параметры, ограничения**
Пользователи	Таблица	Да	Таблица всех пользователей системы. Подробное описание см. ниже.

Таблица «Пользователи»:

Поле	**Тип**	**Ограничения**	**Описание**
Выделить	Check box		Позволяет выделить пользователя, с которым планируется работать далее.
Логин	Text[30]	Только для чтения	Логин пользователя
Имя пользователя	Text[100]	Только для чтения	Фамилия пользователя + Имя пользователя
Статус	Может быть «Новый», «Открытый», «Заблокированный»	Только для чтения	Текущий статус пользователя. Администратор может изменять статус с помощью кнопок «Заблокировать» и «Разблокировать». Описание кнопок – см. ниже.

Кнопки на странице «Пользователи системы»:

Название кнопки	**Тип**	**Ограничения**	**Описание**
Добавить пользователя	Button		Вызывает форму «Информация о пользователе», содержащую поля для деталей нового пользователя
Удалить пользователя	Button	Доступна, только если выделен один или более пользователь	Позволяет удалить выделенного пользователя (пользователей) из системы.
Редактировать пользователя	Button	Доступна, только если выделен один пользователь в таблице «Пользователи»	Вызывает форму «Информация о пользователе», содержащую поля с деталями данного пользователя. Администратор может изменить значение любого из данных пользователя, кроме значения поля «Логин».
Заблокировать	Button	Доступна, только если выделен один или более пользователь	Изменяет статус выделенного пользователя (пользователей) на «Заблокированный».
Разблокировать	Button	Доступна, только если выделен один или более пользователь	Изменяет статус выделенного пользователя (пользователей) на «Открытый».

А.2.1.6. Добавление нового пользователя

Администратор имеет возможность добавлять пользователей в систему. Процесс добавления начинается с нажатия кнопки «Добавить пользователя» на странице «Пользователи системы». (Подробнее об этой странице – см. п.А.2.1.5.) После этого появляется страница «Информация о пользователе». (Подробнее об этой странице – см. п.А.2.1.4.)

Пользователь должен заполнить поля страницы «Информация о пользователе» (все поля обязательны). Для подтверждения создания пользователя необходимо нажать кнопку «Сохранить».

Если такой логин пользователя уже существует, система должна выдать сообщение «*Такой пользователь уже существует в системе. Пожалуйста, введите другой логин*». После этого пользователю предоставляется возможность изменить значение поля «Логин», или же нажать кнопку «Отмена», для отмены создания нового пользователя.

При создании пользователя ему автоматически присваивается статус «Новый», который может быть впоследствии изменен – см. п. А.2.1.5.

А.2.1.7. Удаление пользователя

Администратор имеет право удалять пользователей из системы. При этом из базы данных удаляется вся информация о данном пользователе. Информация о результатах прохождения пользователем тестов из базы данных не удаляется. При необходимости администратор может удалить ее отдельно.

Для удаления пользователя нужно выделить одного или нескольких пользователей в таблице «Пользователи» и нажать кнопку «Удалить пользователя» (см. п. А.2.1.5). После этого система выдает запрос о подтверждении удаления. Если пользователь нажимает кнопку «Да» в диалоге подтверждения, информация о данном пользователе удаляется из БД. Если же пользователь нажимает кнопку «Нет» в диалоге подтверждения, то информация о пользователе не удаляется.

А.2.1.8. Редактирование данных пользователя

Администратор имеет право редактировать данные пользователей системы. Процесс редактирования начинается с нажатия кнопки «Редактировать пользователя» на странице «Пользователи системы». (Подробности - см. таблицы п.А.2.1.5.) Кнопка «Редактировать пользователя» доступна, только если выделен один пользователь в таблице «Пользователи». После этого появляется страница «Информация о пользователе», на которой отображается список полей.

Пользователь может изменить значение полей с данными пользователя кроме поля «Логин». Для подтверждения изменений информации данного пользователя необходимо нажать кнопку «Сохранить». Для отмены изменений необходимо нажать кнопку «Отмена».

А.2.1.9. Просмотр тем тестов

Администратор имеет возможность просмотра всех тем тестов, существующих в системе, в таблице «Тесты в системе», к.т. находится на странице «Тесты». После регистрации в системе клиент тоже получает возможность просматривать темы тестов, но он может просматривать только доступные темы. Поле «Доступно» у клиента видно не будет.

Страница «Тесты»:

Поле	**Тип**	**Обязательное**	**Описание, параметры, ограничения**
Тесты в системе	Таблица	Да	Отображает краткую информацию обо всех тестах системы. Подробное описание см. ниже.

Таблица «Тесты в системе»:

Поле	**Тип**	**Ограничения**	**Описание, параметры, ограничения**
Выделить	Check box		Позволяет выделить тест, с которым планируется работать далее.
Тема	Text[100]	Только для чтения	Название темы теста
Количество вопросов	UINT	Только для чтения	Количество вопросов, которые содержит тест с данной темой.
Доступно	Check box		Признак, закончена ли работа с данным тестом. Если этот флажок установлен в True, то данная тема теста будет видна клиенту при просмотре, и он сможет пройти тест с данной темой.

Кнопки на странице «Тесты»:

Название кнопки	**Тип**	**Ограничения**	**Описание**
Добавить тест	Button		Вызывает форму «Информация о тесте», содержащую поля для деталей новой темы и средства для добавления вопросов.
Удалить тест	Button		Позволяет удалить выделенный тест (тесты) из БД.
Редактировать тест	Button	Доступна только если в таблице «Тесты в системе» выделен один тест и признак «Доступно» для него установлен в false.	Вызывает форму «Информация о тесте», содержащую поля с деталями данного теста.

А.2.1.10. Добавление нового теста

После регистрации в системе под ролью администратора, пользователь получает возможность создавать новые темы тестов. Процесс создания начинается с нажатия кнопки «Добавить тест», которая находится на вкладке «Тесты» под таблицей «Тесты в системе». После этого появляется страница «Информация о тесте», на которой отображается поле «Название темы», а также таблица «Вопросы» к тесту с данной темой (пустая в данном случае). См. таблицы ниже.

Пользователь должен заполнить поле «Название темы». Для подтверждения создания темы необходимо нажать кнопку «Сохранить».

Если тема с таким названием существует, система должна выдать сообщение «*Такая тема уже существует в системе. Пожалуйста, введите другое название темы*». После этого пользователю предоставляется возможность изменить название добавляемой темы, или же нажать кнопку «Отмена», для отмены создания новой темы.

Страница «Информация о тесте»:

Поле	**Тип**	**Обязательное**	**Описание, параметры, ограничения**
Тема	Text[100]	Да	Название новой темы теста. Должно быть уникальным.
Вопросы	Таблица	Да	Таблица вопросов, к.т. содержит тест с данной темой. Подробное описание см. ниже.

Таблица «Вопросы»:

Поле	**Тип**	**Ограничения**	**Описание**
Выделить	Check box		Позволяет выделить вопрос, с которым планируется работать далее.
Вопрос	Text[100]	Только для чтения	Текст вопроса.
Уровень сложности	Может быть «Начальный», «Средний», «Высокий»	Только для чтения	Уровень сложности данного вопроса.
Использовать варианты	Check box	Только для чтения	Признак, будут ли предложены пользователю варианты ответов на данный вопрос. Если он установлен в false, то ответ пользователь должен дать в свободной форме.

Кнопки на странице «Информация о тесте»:

Название кнопки	**Тип**	**Ограничения**	**Описание**
Добавить вопрос	Button		Вызывает форму «Информация о вопросе», содержащую поля для деталей нового вопроса
Удалить вопрос	Button	Доступна только если выделен один или более вопрос в таблице «Вопросы»	Позволяет удалить выделенный вопрос из теста
Редактировать вопрос	Button	Доступна только если выделен один вопрос в таблице «Вопросы»	Вызывает форму «Информация о вопросе», содержащую поля с деталями данного вопроса
Сохранить	Button		Подтверждает изменения на странице «Информация о тесте» и возвращает пользователя на страницу «Темы тестов».
Отмена	Button		Отменяет изменения на странице «Информация о тесте» и возвращает пользователя на страницу «Темы тестов».

А.2.1.11. Удаление теста

Администратор имеет право удалять темы тестов. При этом из базы данных удаляется вся информация о вопросах данной темы, вариантов ответов к этим вопросам. Информация о результатах прохождения пользователями теста с данной темой из базы данных не удаляется. При необходимости администратор может удалить их отдельно.

Для удаления темы пользователь нажимает кнопку «Удалить тему» (см. п. А.2.1.9). После этого система выдает запрос о подтверждении удаления. Если пользователь нажимает кнопку «Да» в диалоге подтверждения, информация о данной теме удаляется из БД. Если же пользователь нажимает кнопку «Нет» в диалоге подтверждения, то тема не удаляется.

А.2.1.12. Просмотр вопросов теста и редактирование теста

Администратор имеет право редактировать информацию тестов. Процесс просмотра (и возможного последующего редактирования) начинается с нажатия кнопки «Редактировать тест», которая находится на странице «Тесты» под таблицей «Тесты в системе». (Подробности о странице «Тесты» - см. таблицы п.А.2.1.9). После этого появляется страница «Информация о тесте», на которой отображается поле «Название темы», а также таблица «Вопросы» к тесту с данной темой. (Подробности о странице «Информация о тесте» - см. таблицы п.А.2.1.10). Таким образом, пользователь может просмотреть вопросы по выбранному тесту.

В процессе редактирования пользователь может изменить значение поля «Название темы», а также изменить состав и содержание вопросов к данному тесту (Подробности об изменении вопросов – см. соответствующие пункты). Для подтверждения изменений данного теста необходимо нажать кнопку «Сохранить». Для отмены изменений необходимо нажать кнопку «Отмена».

Если пользователь изменил название темы теста на название, к.т. уже существует в системе, система должна выдать сообщение «*Такая тема уже существует в системе. Пожалуйста, введите другое название темы*». После этого пользователю предоставляется возможность еще раз изменить название редактируемой темы, или же нажать кнопку «Отмена», для отмены редактирования теста.

А.2.1.13. Добавление вопроса к тесту

Администратор имеет право добавлять новые вопросы к тестам. Процесс создания начинается с нажатия кнопки «Добавить вопрос», которая находится на странице «Информация о тесте». (Подробности о странице «Информация о тесте» - см. таблицы п.А.2.1.10.) После этого появляется страница «Информация о вопросе», на которой отображается список полей – см. таблицы ниже.

Пользователь должен заполнить обязательные поля для вопроса. Для подтверждения создания вопроса необходимо нажать кнопку «Сохранить». Если вопрос использует варианты, а в списке вариантов нет ни одного правильного, выдается сообщение об этом, и пользователю предоставляется возможность добавить правильный вариант ответа в список вариантов.

Пользователь может нажать кнопку «Отмена», для отмены добавления нового вопроса.

В процессе создания нового вопроса пользователь может также создать список вариантов ответов к данному вопросу, если он будет использовать варианты. (Подробности об изменении вариантов ответов – см. соответствующие пункты).

Страница «Информация о вопросе»:

Поле	**Тип**	**Обязательное**	**Описание, параметры, ограничения**
Вопрос	Text[100]	Да	Текст вопроса
Задание	Текст или ссылка	Нет	Также относится к вопросу. Может содержать какое-то предложение, в котором нужно найти ошибки (это будет указано в вопросе).
Уровень сложности	DropDownList. Может быть «Начальный», «Средний», «Высокий»	Да	Уровень сложности добавляемого вопроса. По умолчанию - пустое значение.
Время, сек.	UINT	Да	Максимальное время в секундах, к.т. должно быть затрачено для ответа на данный вопрос. Если при прохождении теста время на вопрос истекает, а ответа не получено, то ответ автоматически считается неправильным.
Варианты ответов	Таблица	Нет	Таблица вариантов ответов. Эта таблица видна, только если флажок «Использовать варианты» установлен в true. Подробное описание – см. ниже.

Таблица «Варианты ответов»:

Поле	**Тип**	**Ограничения**	**Описание**
Выделить	Check box		Позволяет выделить вариант ответа, с которым планируется работать далее.
Вариант ответа	Text[30]	Только для чтения	Текст варианта ответа.
Правильный	Check box	Только для чтения	Признак, является ли этот ответ правильным. В одном вопросе может быть только один правильный вариант ответа.

Кнопки на странице «Информация о вопросе»:

Название кнопки	**Тип**	**Ограничения**	**Описание**
Использовать варианты	Check box		Признак, будут ли предложены пользователю варианты ответов на данный вопрос. По умолчанию установлен в false – это означает, что ответ пользователь должен дать в свободной форме. Если он установлен в true, то становятся видимыми средства управления вариантами ответов на данный вопрос: добавления и удаления вариантов.
Сохранить	Button		Подтверждает изменения на странице «Информация о вопросе» и возвращает пользователя на страницу «Информация о тесте». Если вопрос использует варианты, и в списке вариантов нет ни одного правильного, выдается сообщение об этом, и пользователю предоставляется возможность добавить правильный вариант ответа в список вариантов.
Отмена	Button		Отменяет изменения на странице «Информация о вопросе» и возвращает пользователя на страницу «Информация о тесте».
Добавить вариант	Button	Видна, только если флажок «Использовать варианты» установлен в true.	Делает видимыми поля для нового варианта ответа.
Удалить вариант	Button	Видна, только если флажок «Использовать варианты» установлен в true.	Удаляет выделенный вариант (варианты) ответов.
Да, сохранить	Button	Видна, только если флажок «Использовать варианты» установлен в true.	Сохраняет данный вариант ответа. Поля для добавления нового варианта становятся невидимыми.

Нет, я передумал	Button	Видна, только если флажок «Использовать варианты» установлен в true.	Отменяет добавление нового варианта. Поля для добавления нового варианта становятся невидимыми.

А.2.1.14. Удаление вопроса из теста

Администратор имеет право удалять вопросы из теста. При этом из базы данных удаляется вся информация о вариантах ответов к этому вопросу.

Для удаления вопроса пользователь нажимает кнопку «Удалить вопрос» (см. п. А.2.1.10). После этого система выдает запрос о подтверждении удаления. Если пользователь нажимает кнопку «Да» в диалоге подтверждения, информация о данном вопросе удаляется из БД. Если же пользователь нажимает кнопку «Нет» в диалоге подтверждения, то вопрос не удаляется.

А.2.1.15. Редактирование вопроса

Администратор имеет право редактировать вопросы из теста. Процесс редактирования начинается с нажатия кнопки «Редактировать вопрос», которая находится на странице «Информация о тесте» (Подробности о странице «Информация о тесте» - см. таблицы п.А.2.1.10.) под таблицей «Вопросы». Кнопка «Редактировать вопрос» доступна, только если выделен один вопрос в таблице «Вопросы». После этого появляется страница «Информация о вопросе», на которой отображается список полей. (Подробности о странице «Информация о вопросе» - см. таблицы п.А.2.1.13).

Пользователь может изменить значение полей вопроса, а также изменить состав вариантов ответа к данному вопросу, если он будет использовать варианты. (Подробности об изменении вариантов ответов – см. соответствующие пункты).

Для подтверждения изменений данного вопроса необходимо нажать кнопку «Сохранить». Если вопрос использует варианты, а в списке вариантов нет ни одного правильного, выдается сообщение об этом, и пользователю предоставляется возможность добавить правильный вариант ответа в список вариантов.

Для отмены изменений необходимо нажать кнопку «Отмена».

А.2.1.16. Добавление варианта ответа к вопросу

Администратор имеет право добавлять варианты ответов к вопросам, которые используют варианты. (Чтобы вопрос использовал варианты, нужно установить в true флажок «Использовать варианты» на странице «Информация о вопросе».) Процесс добавления начинается с нажатия кнопки «Добавить вариант», которая находится на странице «Информация о вопросе». (Подробности о странице «Информация о вопросе» - см. таблицы п.А.2.1.13.) После этого под таблицей «Варианты ответов» появляются поля для нового варианта ответов – см. таблица ниже.

Пользователь должен заполнить обязательное поле «Вариант ответа», и при необходимости установить флажок «Правильный». Для подтверждения создания варианта ответа необходимо нажать кнопку «Да, сохранить», которая находится под полями для нового варианта. Для отмены добавления нового варианта пользователь может нажать кнопку «Нет, я передумал».

А.2.1.17. Удаление варианта ответа

Администратор имеет право удалять варианты ответа к вопросу из списка вариантов. При этом из базы данных удаляется вся информация о данном варианте ответа к данному вопросу.

Для удаления варианта пользователь нажимает кнопку «Удалить вариант» (см. п. А.2.1.13). После этого система выдает запрос о подтверждении удаления. Если пользователь нажимает кнопку «Да» в диалоге подтверждения, информация о данном варианте удаляется из БД. Если же пользователь нажимает кнопку «Нет» в диалоге подтверждения, то вариант не удаляется.

А.2.1.18. Прохождение теста

Клиент, который зарегистрирован в системе и имеет статус «Открытый» имеет право выбрать любую тему из доступных клиентам тем, и пройти по ней тест. Для этого клиент должен выделить любой тест на странице «Тесты в системе» и нажать кнопку «Пройти тест». После этого появляется краткая страница общей информации о том, как построен тест. Для начала непосредственного прохождения теста нужно нажать кнопку «Начать». После этого появится окно, содержащее первый вопрос теста. На этом окне будет отображаться счетчик времени, показывающий оставшееся время в секундах для ответа на данный вопрос. Если это время истекает, а ответ от пользователя не получен, то ответ на данный вопрос автоматически считается неправильным, а пользователя переводят на следующий вопрос.

По окончании прохождения клиентом теста система выдает ему для просмотра результаты в виде полного списка, содержащего номер вопроса (без текста вопроса), уровень его сложности, и отметку, правильным ли был ответ. Также ставится определенный средний балл за все задания, содержащие варианты. Оно автоматически заносится в базу данных. Ответы же на вопросы, которые давались в свободной форме, преподаватель проверяет вручную, и вручную же заносит их в базу данных.

А.2.2. Требования к производительности

Требований к производительности не имеется.

Приложение Б

Образец оформления титульного листа

Белорусский национальный технический университет
Кафедра ПОВТ и АС

КОНТРОЛЬНАЯ РАБОТА

по дисциплине «ТЕСТИРОВАНИЕ И ОТЛАДКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ»

выполнил ст. гр. 307210 Иванов И.И.

проверила Попова Ю.Б.

Минск 2010

Приложение В

Содержание отчета

Отчет по контрольной работе должен содержать:

1. Номер задания.
2. Название задания.
3. Постановка задачи.
4. Текст выполненного задания.
5. Выводы по заданию.
6. Список используемой литературы.
7. Приложение с листингом исходного кода тестируемой программы.